```
FFFFFFFFFFFFFFF     AAAAAAAAA     LLL
FFFFFFFFFFFFFFF     AAAAAAAAA     LLL
FFFFFFFFFFFFFFF     AAAAAAAAA     LLL
FFF              AAA         AAA  LLL
FFF              AAA         AAA  LLL
FFF              AAA         AAA  LLL
FFF              AAA         AAA  LLL
FFF              AAA         AAA  LLL
FFF              AAA         AAA  LLL
FFFFFFFFFFFF     AAA         AAA  LLL
FFFFFFFFFFFF     AAA         AAA  LLL
FFFFFFFFFFFF     AAA         AAA  LLL
FFF              AAAAAAAAAAAAAAA  LLL
FFF              AAAAAAAAAAAAAAA  LLL
FFF              AAAAAAAAAAAAAAA  LLL
FFF              AAA         AAA  LLL
FFF              AAA         AAA  LLL
FFF              AAA         AAA  LLL
FFF              AAA         AAA  LLLLLLLLLLLLLLL
FFF              AAA         AAA  LLLLLLLLLLLLLLL
FFF              AAA         AAA  LLLLLLLLLLLLLLL
```

```
_$a

Val
---
000
000
000
000
000
000
000
000
000
000
000
000
000
000
000
000
000
000
000
000
000
000
000
000
7FF
7FF
7FF
7FF
7FF
7FF
7FF
7FF
7FF
7FF
7FF
7FF
7FF
7FF
7FF
7FF
7FF
7FF
7FF
7FF
7FF
7FF
7FF
7FF
7FF
7FF
7FF
7FF
7FF
7FF
7FF
7FF
```

```
**FILE**ID**FALACTMSG                                M 9
```

```
FFFFFFFFFF    AAAAAA    LL            AAAAAA      CCCCCCCC  TTTTTTTTTT  MM      MM    SSSSSSSS    GGGGGGGG
FFFFFFFFFF    AAAAAA    LL            AAAAAA      CCCCCCCC  TTTTTTTTTT  MM      MM    SSSSSSSS    GGGGGGGG
FF          AA      AA  LL          AA      AA  CC              TT      MMMM  MMMM  SS          GG
FF          AA      AA  LL          AA      AA  CC              TT      MMMM  MMMM  SS          GG
FF          AA      AA  LL          AA      AA  CC              TT      MM  MM  MM  SS          GG
FF          AA      AA  LL          AA      AA  CC              TT      MM  MM  MM  SS          GG
FFFFFFFF    AA      AA  LL          AA      AA  CC              TT      MM      MM    SSSSSS    GG
FFFFFFFF    AA      AA  LL          AA      AA  CC              TT      MM      MM    SSSSSS    GG
FF          AAAAAAAAAA  LL          AAAAAAAAAA  CC              TT      MM      MM          SS  GG  GGGGGG
FF          AAAAAAAAAA  LL          AAAAAAAAAA  CC              TT      MM      MM          SS  GG  GGGGGG
FF          AA      AA  LL          AA      AA  CC              TT      MM      MM          SS  GG      GG        ....
FF          AA      AA  LL          AA      AA  CC              TT      MM      MM          SS  GG      GG        ....
FF          AA      AA  LLLLLLLLLL  AA      AA    CCCCCCCC      TT      MM      MM  SSSSSSSS      GGGGGG          ....
FF          AA      AA  LLLLLLLLLL  AA      AA    CCCCCCCC      TT      MM      MM  SSSSSSSS      GGGGGG          ....


LL            IIIIII      SSSSSSSS
LL            IIIIII      SSSSSSSS
LL              II      SS
LL              II      SS
LL              II      SS
LL              II      SS
LL              II        SSSSSS
LL              II        SSSSSS
LL              II              SS
LL              II              SS
LL              II              SS
LL              II              SS
LLLLLLLLLL    IIIIII    SSSSSSSS
LLLLLLLLLL    IIIIII    SSSSSSSS
```

Table of contents

FAL
V04

```
0000            .TITLE  FALACTMSG - STATE TABLE ACTION ROUTINES
0000            .IDENT  'V04-000'
0000
0000    ;
0000    ;*****************************************************************************
0000    ;*                                                                           *
0000    ;*  COPYRIGHT (c) 1978, 1980, 1982, 1984 BY                                  *
0000    ;*  DIGITAL EQUIPMENT CORPORATION, MAYNARD, MASSACHUSETTS.                   *
0000    ;*  ALL RIGHTS RESERVED.                                                     *
0000    ;*                                                                           *
0000    ;*  THIS SOFTWARE IS FURNISHED UNDER A LICENSE AND MAY BE USED AND COPIED    *
0000    ;*  ONLY IN  ACCORDANCE WITH  THE  TERMS  OF  SUCH  LICENSE  AND WITH THE    *
0000    ;*  INCLUSION OF THE ABOVE COPYRIGHT NOTICE. THIS SOFTWARE OR  ANY  OTHER    *
0000    ;*  COPIES THEREOF MAY NOT BE PROVIDED OR OTHERWISE MADE AVAILABLE TO ANY    *
0000    ;*  OTHER PERSON.  NO TITLE TO AND OWNERSHIP OF  THE  SOFTWARE IS  HEREBY    *
0000    ;*  TRANSFERRED.                                                             *
0000    ;*                                                                           *
0000    ;*  THE INFORMATION IN THIS SOFTWARE IS  SUBJECT TO CHANGE WITHOUT NOTICE    *
0000    ;*  AND  SHOULD  NOT  BE  CONSTRUED AS  A COMMITMENT BY DIGITAL EQUIPMENT    *
0000    ;*  CORPORATION.                                                             *
0000    ;*                                                                           *
0000    ;*  DIGITAL ASSUMES NO RESPONSIBILITY FOR THE USE  OR  RELIABILITY OF ITS    *
0000    ;*  SOFTWARE ON EQUIPMENT WHICH IS NOT SUPPLIED BY DIGITAL.                  *
0000    ;*                                                                           *
0000    ;*                                                                           *
0000    ;*****************************************************************************
0000    ;
0000
0000    ;++
0000    ; Facility: FAL (DECnet File Access Listener)
0000    ;
0000    ; Abstract:
0000    ;
0000    ;       This module contains action routines called by the state table manager.
0000    ;
0000    ; Environment: VAX/VMS, user mode
0000    ;
0000    ; Author: James A. Krycka,      Creation Date:  16-JUN-1977
0000    ;
0000    ; Modified By:
0000    ;
0000    ;       V03-007 JAK0136         J A Krycka      07-MAR-1984
0000    ;               Support FAL logging options that deal with fields in the DAP
0000    ;               Configuration message sent to partner.
0000    ;
0000    ;       V03-006 JAK0118         J A Krycka      12-JUL-1983
0000    ;               Fix bug in processing the DAP KEY field.
0000    ;
0000    ;       V03-005 KRM0104         K Malik         10-May-1983
0000    ;               Update symbols to match Dap V7.0 spec.
0000    ;
0000    ;       V03-004 JAK0104         J A Krycka      29-APR-1983
0000    ;               Make minor enhancements to FAL logging display.
0000    ;
0000    ;       V03-003 KRM0083         K Malik         23-Mar-1983
0000    ;               Add support for STMLF and STMCR formats.
0000    ;
```

```
                0000    58 ;        V03-002 KRM0074         K Malik         23-Nov-1982
                0000    59 ;                Added FAL$DECODE_NAM routine (to support $RENAME function).
                0000    60 ;
                0000    61 ;        V03-001 JAK0101         J A Krycka      09-OCT-1982
                0000    62 ;                Fix bug in converting DAP OWNER value into binary format.
                0000    63 ;
                0000    64 ;--
```

```
0000            .SBTTL  DECLARATIONS
0000
0000    ;
0000    ; Include Files:
0000    ;
0000
0000            $DAPPLGDEF                              ; Define DAP prologue symbols
0000            $DAPHDRDEF                              ; Define DAP message header
0000            $DAPSSPDEF                              ; Define DAP system specific field
0000            $DAPCNFDEF                              ; Define DAP Configuration message
0000            $DAPATTDEF                              ; Define DAP Attributes message
0000            $DAPACCDEF                              ; Define DAP Access message
0000            $DAPCTLDEF                              ; Define DAP Control message
0000            $DAPCONDEF                              ; Define DAP Continue Transfer message
0000            $DAPCMPDEF                              ; Define DAP Access Complete message
0000            $DAPKEYDEF                              ; Define DAP Key Definition message
0000            $DAPALLDEF                              ; Define DAP Allocation message
0000            $DAPTIMDEF                              ; Define DAP Date and Time message
0000            $DAPPRODEF                              ; Define DAP Protection message
0000            $DAPNAMDEF                              ; Define DAP Name message
0000            $DEVDEF                                 ; Define Device Characteristics symbols
0000            $FABDEF                                 ; Define File Access Block symbols
0000            $FALWRKDEF                              ; Define FAL Work Area symbols
0000            $RABDEF                                 ; Define Record Access Block sym**
0000            $XABDEF                                 ; Define symbols common to all XABs
0000            $XABALLDEF                              ; Define Allocation XAB symbols
0000            $XABDATDEF                              ; Define Date and Time XAB symbols
0000            $XABKEYDEF                              ; Define Key Definition XAB symbols
0000            $XABPRODEF                              ; Define Protection XAB symbols
0000    ;       $XABRDTDEF                              ; Define Revision Date and Time symbols
0000
0000    ;
0000    ; Macros:
0000    ;
0000    ;       None
0000    ;
0000    ; Equated Symbols:
0000    ;
0000    ;
0000
0000            ASSUME  DAP$Q_DCODE_FLG EQ 0
0000            ASSUME  FAL$Q_FLG EQ 0
0000
0000    ;
0000    ; Own Storage:
0000    ;
```

```
    0000          .SBTTL  ACTION ROUTINES
00000000          .PSECT  FAL$CODE        NOSHR,EXE,RD,NOWRT,BYTE
    0000
    0000    ;++
    0000    ; Functional Description:
    0000    ;
    0000    ;       This module contains action routines invoked by the state table
    0000    ;       manager (FAL$STATE).
    0000    ;
    0000    ;       The input parameters and completion codes listed below are applicable
    0000    ;       for all of these action routines. Note that an action routine may use
    0000    ;       R0-R7 and AP without restoring them on exit. R0 on exit, however, must
    0000    ;       represent a status code to indicate success/failure of the routine or
    0000    ;       a true/false condition, as appropriate. This status code is used by
    0000    ;       the state table manager to advance to the next state.
    0000    ;
    0000    ; Calling Sequence:
    0000    ;
    0000    ;       BSBW    FAL$name
    0000    ;
    0000    ; Input Parameters:
    0000    ;
    0000    ;       R8      Address of FAL work area
    0000    ;       R9      Address of DAP control block
    0000    ;       R10     Address of FAB
    0000    ;       R11     Address of RAB
    0000    ;
    0000    ; Implicit Inputs:
    0000    ;
    0000    ;       None
    0000    ;
    0000    ; Output Parameters:
    0000    ;
    0000    ;       R0      Status code
    0000    ;       R1-R7   Destroyed
    0000    ;       AP      Destroyed
    0000    ;
    0000    ; Implicit Outputs:
    0000    ;
    0000    ;       None
    0000    ;
    0000    ; Completion Codes:
    0000    ;
    0000    ;       R0      1 = success; 0 = failure
    0000    ;
    0000    ; Side Effects:
    0000    ;
    0000    ;       None
    0000    ;
    0000    ;--
```

```
                          0000          .SBTTL  FAL$DECODE_CNF
                          0000
                          0000  ;++
                          0000  ; Process the Configuration message which has been received and validated.
                          0000  ; Return a Configuration message to partner and determine the DAP buffer size
                          0000  ; to use which is the smaller of partner's buffer size and FAL's buffer size.
                          0000  ;--
                          0000
                          0000  FAL$DECODE_CNF::                           ; Entry point
                          0000          $SETBIT #FAL$V_CNF_MSG,(R8)        ; Denote Configuration message received
                          0004          $CLRBIT #FAL$V_ATT_MSG,(R8)        ;  and discard any previous Attributes
                          0008                                             ;  message
      57    18 A8    3C   0008          MOVZWL  FAL$W_QIOBUFSIZ(R8),R7     ; Get FAL's buffer size (i.e., largest
                          000C                                             ;  I/O buffer size supported by process)
         68    39    E1   000C          BBC     #FAL$V_USE_DBS,(R8),-      ; Branch to use calculated buffer size
               05         000F                  SEND_CNF                   ;
      57  00A0 C8    3C   0010          MOVZWL  FAL$W_USE_DBS(R8),R7       ; Override with user specified value
                          0015
                          0015  ;+
                          0015  ; Build and send Configuration message to partner.
                          0015  ;-
                          0015
                          0015  SEND_CNF:                                  ;
                          0015          $SETBIT #FAL$V_LAST_MSG,(R8)       ; Declare this last message to block
         50    01    D0   0019          MOVL    #DAP$K_CNF_MSG,R0          ; Get message type value
             FFE1'   30   001C          BSBW    FAL$BUILD_HEAD             ; Construct message header
         83    57    B0   001F          MOVW    R7,(R3)+                   ; Store BUFSIZ field
         83    07    90   0022          MOVB    #DAP$K_VAXVMS,(R3)+        ; Store OSTYPE field
         83    03    90   0025          MOVB    #DAP$K_RMS32,(R3)+         ; Store FILESYS field
      06 68    3A    E1   0028          BBC     #FAL$V_USE_SYS,(R8),2$     ; Branch to use standard values
   FE A3  00A2 C8    B0   002C          MOVW    FAL$W_USE_SYS(R8),-2(R3)   ; Override with user specified values
         83    07    90   0032  2$:     MOVB    #DAP$K_VERNUM_V,(R3)+      ; Store VERNUM field
         83    00    90   0035          MOVB    #DAP$K_ECONUM_V,(R3)+      ; Store ECONUM field
         83    C0    90   0038          MOVB    #DAP$K_USRNUM_V,(R3)+      ; Store USRNUM field
         83    04    90   003B          MOVB    #DAP$K_DECVER_V,(R3)+      ; Store DECVER field
      06 68    3B    E1   003E          BBC     #FAL$V_USE_VER,(R8),4$     ; Branch to use standard values
   FC A3  00A4 C8    D0   0042          MOVL    FAL$L_USE_VER(R8),-4(R3)   ; Override with user specified values
         83    00    90   0048  4$:     MOVB    #DAP$K_USRVER_V,(R3)+      ; Store USRVER field
                          004B
                          004B  ;
                          004B  ; Construct the system capabilities field.
                          004B  ; Also, check the debugging options to disable message blocking and DAP level
                          004B  ; CRC checking (after any user specified system capabilites bitmasks, if any,
                          004B  ; have been applied).
                          004B  ;
                          004B
51     EFF67DF7 8F   D0   004B          MOVL    #DAP$K_SYSCAP1_V,R1        ; Get VAX supported capabilities
52     00001962 8F   D0   0052          MOVL    #DAP$K_SYSCAP2_V,R2        ;  quadword bitmask
                          0059                                             ; ------ process debugging options -----
      05 68    3C    E1   0059          BBC     #FAL$V_USE_SC1,(R8),6$     ; Branch to use standard values
      51  00A8 C8    D0   005D          MOVL    FAL$L_USE_SC1(R8),R1       ; Override with user specified values
      05 68    3D    E1   0062  6$:     BBC     #FAL$V_USE_SC2,(R8),8$     ; Branch to use standard values
      52  00AC C8    D0   0066          MOVL    FAL$L_USE_SC2(R8),R2       ; Override with user specified values
      0F 68    31    E1   006B  8$:     BBC     #FAL$V_DIS_MBK,(R8),10$    ; Is DAP message blocking disabled?
51     00140000 8F   CA   006F          BICL2   #<<1@DAP$V_MSGBLK>!-       ; Yes, clear message blocking bits in
                          0076                     <1@DAP$V_BIGBLK>!-      ;  system capabilities bitmask for
                          0076                     0>,R1                   ;  Configuration message to transmit
```

```
       28 A9   00140000 8F   CA  0076          BICL2   #<<1@DAP$V_MSGBLK>!-       ; Also, clear message blocking bits in
                                 007E                     <1@DAP$V_BIGBLK>!-        ;  system capabilities bitmask
                                 007E                   0>,DAP$Q_SYSCAP(R9)        ;  received from partner
                 09 68    30  E1 007E   10$:   BBC     #FAL$V_DIS_CRC,(R8),20$    ; Is file level CRC checksum disabled?
                                 0082          $CLRBIT #DAP$V_DAPCRC,R1            ; Yes, clear bits in both XMT and RCV
                                 0086          $CLRBIT #DAP$V_DAPCRC,-             ;  system capabilities fields
                                 0086                  DAP$Q_SYSCAP(R9)            ;
                                 008B                                              ; ------ finish debugging options -----
                      FF72'   30 008B   20$:   BSBW    FAL$CVT_BN8_EXT             ; Store SYSCAP as an extensible field
                      FF6F'   30 008E          BSBW    FAL$BUILD_TAIL              ; Finish building message
                      FF6C'   30 0091          BSBW    FAL$TRANSMIT                ; Send Configuration message
                                 0094
                                 0094   ;+
                                 0094   ; Determine the 'agreed upon' DAP buffer size to use and save this value.
                                 0094   ; It is the smaller of partner's buffer size and FAL's maximum buffer size.
                                 0094   ;-
                                 0094
                      40 A9   B0 0094          MOVW    DAP$W_BUFSIZ(R9),-          ; Assume we'll use partner's
                      1A A8      0097                  FAL$W_DAPBUFSIZ(R8)         ;  buffer size
                         06   13 0099          BEQL    30$                         ; Branch if partner has unlimited space
                 57   40 A9   B1 009B          CMPW    DAP$W_BUFSIZ(R9),R7         ; Compare partner's buffer size with
                                 009F                                              ;  our buffer size
                         04   1B 009F          BLEQU   40$                         ; Branch if partner has less capacity
                 1A A8   57   B0 00A1   30$:   MOVW    R7,FAL$W_DAPBUFSIZ(R8)      ; We guessed wrong, so we'll use
                                 00A5                                              ;  our buffer size
                       04E4   31 00A5   40$:   BRW     EXIT_SUCCESS                ; Exit state with success
```

```
                         00A8         .SBTTL  FAL$DECODE_ATT
                         00A8
                         00A8   ;++
                         00A8   ; Process the Attributes message which has been received and validated.
                         00A8   ; Update the FAB and FHCXAB with information from this message.
                         00A8   ;--
                         00A8
                         00A8   FAL$DECODE_ATT::                               ; Entry point
                         00A8
                         00A8         $SETBIT #FAL$V_ATT_MSG,(R8)      ; Denote Attributes message received
                         00AC
                         00AC   ;
                         00AC   ; Save the DAP DATATYPE field for use later.
                         00AC   ;
                         00AC
01F4 C8    44 A9     90  00AC         MOVB    DAP$B_DATATYPE(R9),FAL$B_DATATYPE(R8)
                         00B2
                         00B2   ;
                         00B2   ; Process the DAP ORG, RFM and RAT fields.
                         00B2   ;
                         00B2
                         00B2         ASSUME  DAP$K_SEQ EQ FAB$C_SEQ
                         00B2         ASSUME  DAP$K_REL EQ FAB$C_REL
                         00B2         ASSUME  DAP$K_IDX EQ FAB$C_IDX
                         00B2
  1D AA    45 A9     90  00B2         MOVB    DAP$B_ORG(R9),FAB$B_ORG(R10)
                         00B7
                         00B7         ASSUME  DAP$K_UDF EQ FAB$C_UDF
                         00B7         ASSUME  DAP$K_FIX EQ FAB$C_FIX
                         00B7         ASSUME  DAP$K_VAR EQ FAB$C_VAR
                         00B7         ASSUME  DAP$K_VFC EQ FAB$C_VFC
                         00B7         ASSUME  DAP$K_STM EQ FAB$C_STM
                         00B7         ASSUME  DAP$K_STMLF EQ FAB$C_STMLF
                         00B7         ASSUME  DAP$K_STMCR EQ FAB$C_STMCR
                         00B7
  1F AA    46 A9     90  00B7         MOVB    DAP$B_RFM(R9),FAB$B_RFM(R10)
                         00BC
                         00BC         ASSUME  DAP$V_FTN EQ FAB$V_FTN
                         00BC         ASSUME  DAP$V_CR  EQ FAB$V_CR
                         00BC         ASSUME  DAP$V_PRN EQ FAB$V_PRN
                         00BC         ASSUME  DAP$V_BLK EQ FAB$V_BLK
                         00BC
  1E AA    47 A9     90  00BC         MOVB    DAP$B_RAT(R9),FAB$B_RAT(R10)
              10     8A  00C1         BICB2   #DAP$M_EMBEDDED,-         ; Ignore this bit
           1E AA         00C3                 FAB$B_RAT(R10)            ;
   0A 69      34     E0  00C5         BBS     #DAP$V_VAXVMS,(R9),10$    ; Branch if partner is VAX/VMS
   04      46 A9     91  00C9         CMPB    DAP$B_RFM(R9),#DAP$K_STM; Branch if not stream format
              04     12  00CD         BNEQ    10$                       ;
   1E AA      02     90  00CF         MOVB    #FAB$M_CR,FAB$B_RAT(R10); If it is, declare cc to be implied
                         00D3
                         00D3   ;
                         00D3   ; Process the DAP BLS, MRS, ALQ, BKS, FSZ, MRN, and DEQ fields.
                         00D3   ;
                         00D3
  3C AA    48 A9     B0  00D3   10$:  MOVW    DAP$W_BLS(R9),FAB$W_BLS(R10)
  36 AA    4A A9     B0  00D8         MOVW    DAP$W_MRS(R9),FAB$W_MRS(R10)
  10 AA    4C A9     D0  00DD         MOVL    DAP$L_ALQ1(R9),FAB$L_ALQ(R10)
```

```
   3E AA    50 A9   90  00E2          MOVB    DAP$B_BKS(R9),FAB$B_BKS(R10)
   3F AA    51 A9   90  00E7          MOVB    DAP$B_FSZ(R9),FAB$B_FSZ(R10)
   38 AA    58 A9   D0  00EC          MOVL    DAP$L_MRN(R9),FAB$L_MRN(R10)
   14 AA    54 A9   B0  00F1          MOVW    DAP$W_DEQ1(R9),FAB$W_DEQ(R10)
                        00F6
                        00F6    ;
                        00F6    ; Process the DAP FOP field after saving it for use later.
                        00F6    ;
                        00F6
        51    64 A9   D0  00F6          MOVL    DAP$L_FOP1(R9),R1          ; Get DAP FOP bits and
   01F8 C8      51   D0  00FA          MOVL    R1,FAL$L_FOP(R8)           ;  save field for use later
              0342   30  00FF          BSBW    MAP_FOP_FIELD              ; Update FOP in FAB
                        0102
                        0102    ;
                        0102    ; Process the DAP LRL field.
                        0102    ; This is the only FHCXAB field that is input to RMS, and then only for the
                        0102    ; $CREATE function where the record format is variable or VFC.
                        0102    ;
                        0102
              70 A9   B0  0102          MOVW    DAP$W_LRL(R9),-            ; Copy value to FHCXAB
           02F4'C8        0105                  FAL$L_FHCXAB+XAB$W_LRL(R8)
              0481   31  0108          BRW     EXIT_SUCCESS               ; Exit state with success
```

```
                            010B          .SBTTL  FAL$DECODE_ACC
                            010B
                            010B    ;++
                            010B    ; Process the Access message which has been received and validated.
                            010B    ; Update the FAB with information from this message.
                            010B    ;--
                            010B
                            010B    FAL$DECODE_ACC::                                ; Entry point
                            010B
                            010B    ;
                            010B    ; Save the DAP ACCFUNC, ACCOPT, and DISPLAY fields for use later.
                            010B    ;
                            010B
 01F6 C8    40 A9     90    010B          MOVB    DAP$B_ACCFUNC(R9),FAL$B_ACCFUNC(R8)
 01F5 C8    41 A9     90    0111          MOVB    DAP$B_ACCOPT(R9),FAL$B_ACCOPT(R8)
   70 A8    4C A9     B0    0117          MOVW    DAP$W_DISPLAY1(R9),FAL$W_DISPLAY(R8)
                            011C
                            011C    ;
                            011C    ; Process the DAP file specification field.
                            011C    ;
                            011C
            44 A9     90    011C          MOVB    DAP$Q_FILESPEC(R9),-       ; Store size of filespec string
            34 AA           011F                  FAB$B_FNS(R10)             ;  in FAB
            44 A9     28    0121          MOVC3   DAP$Q_FILESPEC(R9),-       ; Copy filespec string to buffer
            48 B9           0124                  @DAP$Q_FILESPEC+4(R9),-    ;
            2C BA           0126                  @FAB$L_FNA(R10)
       51   40 A9     9A    0128          MOVZBL  DAP$B_ACCFUNC(R9),R1       ; Get access function code
       52   44 A9     7E    012C          MOVAQ   DAP$Q_FILESPEC(R9),R2      ; Get address of filename descriptor
            FECD'     30    0130          BSBW    FAL$LOG_REQNAM             ; Log requested name in print file
                            0133
                            0133    ;
                            0133    ; Process the DAP FAC field.
                            0133    ;
                            0133
                            0133          ASSUME  DAP$V_PUT EQ FAB$V_PUT
                            0133          ASSUME  DAP$V_GET EQ FAB$V_GET
                            0133          ASSUME  DAP$V_DEL EQ FAB$V_DEL
                            0133          ASSUME  DAP$V_UPD EQ FAB$V_UPD
                            0133          ASSUME  DAP$V_TRN EQ FAB$V_TRN
                            0133          ASSUME  DAP$V_BIO EQ FAB$V_BIO
                            0133          ASSUME  DAP$V_BRO EQ FAB$V_BRO
                            0133          ASSUME  DAP$V_APP EQ FAB$V_EXE   ; Map APP to PUT
                            0133
   16 AA    42 A9     90    0133          MOVB    DAP$B_FAC(R9),FAB$B_FAC(R10)
 05 16 AA      07     E5    0138          BBCC    #FAB$V_EXE,FAB$B_FAC(R10),10$
 00 16 AA      00     E2    013D          BBSS    #FAB$V_PUT,FAB$B_FAC(R10),10$
                            0142
                            0142    ;
                            0142    ; Process the DAP SHR field.
                            0142    ;
                            0142
               52     D4    0142    10$:  CLRL    R2                         ; Clear RMS SHR bits
       51   43 A9     9A    0144          MOVZBL  DAP$B_SHR(R9),R1           ; Get DAP SHR bits
               30     13    0148          BEQL    20$                        ; Branch if no bits to map
                            014A          $MAPBIT DAP$V_SHRPUT,FAB$V_SHRPUT; Map SHRPUT bit
                            0152          $MAPBIT DAP$V_SHRGET,FAB$V_SHRGET; Map SHRGET bit
                            015A          $MAPBIT DAP$V_SHRDEL,FAB$V_SHRDEL; Map SHRDEL bit
```

```
                      0162              $MAPBIT DAP$V_SHRUPD,FAB$V_SHRUPD; Map SHRUPD bit
                      016A              $MAPBIT DAP$V_UPI,FAB$V_UPI     ;  Map UPI bit
                      0172              $MAPBIT DAP$V_NIL,FAB$V_NIL     ;  Map NIL bit
 17 AA    52    90    017A   20$:       MOVB    R2,FAB$B_SHR(R10)       ; Update SHR field in FAB
                      017E
                      017E   ;
                      017E   ; Use the ACCFUNC field value as the next state table value.
                      017E   ;
                      017E
    40 A9       90    017E              MOVB    DAP$B_ACCFUNC(R9),-     ; Store new state transition value
    10 A8             0181                      FAL$B_VALUE(R8)         ;
       00       E1    0183              BBC     #DAP$V_LOAD,-           ; Branch if no system specific
 0A 0084 C9           0185                      DAP$L_SSP_FLG(R9),30$   ;  function modifier found
                      0189              $SETBIT #FAB$V_SQO,FAB$L_FOP(R10)
                      018E                                              ; Force sequential file transfer
                      018E                                              ;  mode for efficiency
    FF 8F       90    018E              MOVB    #DAP$K_LOAD,-           ; Make new state transition value
    10 A8             0191                      FAL$B_VALUE(R8)         ;  the load image function
     03F6       31    0193   30$:       BRW     EXIT_SUCCESS            ; Exit state with success
```

```
                    0196           .SBTTL  FAL$DECODE_CTL
                    0196
                    0196   ;++
                    0196   ; Process the Control message which has been received and validated.
                    0196   ; Update the RAB with information from this message.
                    0196   ;--
                    0196
                    0196   FAL$DECODE_CTL::                              ; Entry point
                    0196
                    0196   ;+
                    0196   ; Save the DAP DISPLAY field for use later if we're not in a wildcard context.
                    0196   ; In wildcard file retrieval, for example, the DAP Access message is sent only
                    0196   ; once, thus FAL$W_DISPLAY must reflect the DISPLAY value from the Access
                    0196   ; message on subsequent file opens. Since the Control message functions of
                    0196   ; DISPLAY and EXTEND are not valid in a wildcard context (which require
                    0196   ; FAL$W_DISPLAY to be updated), this special check is an acceptible solution
                    0196   ; to a wildcard retrieval problem.
                    0196   ;-
                    0196
   68   0A    E0    0196           BBS     #FAL$V_WILD,(R8),-       ; Branch if wildcard operation
        05          0199                   RAC_FIELD                ;
   54   A9    B0    019A           MOVW    DAP$W_DISPLAY2(R9),-     ; Save display message bitmask in
   70   A8          019D                   FAL$W_DISPLAY(R8)        ;  FAL work area
                    019F
                    019F   ;+
                    019F   ; Process the DAP RAC field.
                    019F   ; In addition to normal RMS-32 RAC information, this field specifies whether
                    019F   ; the access is to be in:
                    019F   ;    (1) file transfer mode or record transfer mode
                    019F   ;    (2) block I/O mode or record I/O mode
                    019F   ;
                    019F   ; Note: If the RAC field is not present in the Control message, then the default
                    019F   ;       action is to use the previous value.
                    019F   ;-
                    019F
                    019F           ASSUME  DAP$K_SEQ_ACC  EQ 0
                    019F           ASSUME  DAP$K_KEY_ACC  EQ 1
                    019F           ASSUME  DAP$K_RFA_ACC  EQ 2
                    019F           ASSUME  DAP$K_SEQ_FILE EQ 3
                    019F           ASSUME  DAP$K_BLK_VBN  EQ 4
                    019F           ASSUME  DAP$K_BLK_FILE EQ 5
                    019F
                    019F   RAC_FIELD:                               ; Process record access field
        00    E0    019F           BBS     #DAP$V_RAC,-             ; Branch if RAC field was explicitly
06 44   A9          01A1                   DAP$W_CTLMENU(R9),10$    ;  specified
 01F7   C8    90    01A4           MOVB    FAL$B_RAC(R8),-          ; If not, use previous value saved in
   46   A9          01A8                   DAP$B_RAC(R9)            ;  FAL work area
   46   A9    90    01AA   10$:    MOVB    DAP$B_RAC(R9),-          ; Save currently specified value as
 01F7   C8          01AD                   FAL$B_RAC(R8)            ;  previous value for next-time-thru
                    01B0           $CASEB  SELECTOR=DAP$B_RAC(R9)-  ; Dispatch on DAP record access mode:
                    01B0                   DISPL=<-
                    01B0                           20$-             ;  Sequential record access
                    01B0                           20$-             ;  Random access by key value
                    01B0                           20$-             ;  Random access by RFA
                    01B0                           30$-             ;  Sequential file transfer
                    01B0                           40$-             ;  Block I/O access by VBN
                    01B0                           50$-             ;  Block I/O sequential file transfer
```

```
                             01B0                  >                        ;
                             01C1      ;
                             01C1      ; Update the RAC field of the RAB unless block I/O mode is specified.
                             01C1      ; (RMS-32 ignores the RAC field on block I/O operations.)
                             01C1      ;
                             01C1      ; Also update the file transfer mode and block I/O flags as appropriate
                             01C1      ; for the access mode invoked.
                             01C1      ;
                             01C1
                             01C1              ASSUME  DAP$K_SEQ_ACC EQ RAB$C_SEQ
                             01C1              ASSUME  DAP$K_KEY_ACC EQ RAB$C_KEY
                             01C1              ASSUME  DAP$K_RFA_ACC EQ RAB$C_RFA
                             01C1
            46 A9      90    01C1      20$:    MOVB    DAP$B_RAC(R9),-          ; Store record access mode in RAB
               1E AB         01C4                      RAB$B_RAC(R11)           ;  (either SEQ, KEY, or RFA)
                             01C6              $CLRBIT #FAL$V_FTM,(R8)          ; Say record transfer mode
                  08   11    01CA              BRB     35$                      ;
      1E AB       00   90    01CC      30$:    MOVB    #RAB$C_SEQ,RAB$B_RAC(R11);Set record access mode to SEQ in RAB
                             01D0              $SETBIT #FAL$V_FTM,(R8)          ; Say file transfer mode
                             01D4      35$:    $CLRBIT #FAL$V_BLK_IO,(R8)       ; Say record I/O mode
                  0E   11    01D8              BRB     ROP_FIELD                ;
                             01DA      40$:    $CLRBIT #FAL$V_FTM,(R8)          ; Say record transfer mode
                  04   11    01DE              BRB     55$                      ;
                             01E0      50$:    $SETBIT #FAL$V_FTM,(R8)          ; Say file transfer mode
                             01E4      55$:    $SETBIT #FAL$V_BLK_IO,(R8)       ; Say block I/O mode
                             01E8
                             01E8      ;+
                             01E8      ; Process the DAP ROP field.
                             01E8      ;
                             01E8      ; Note: If the ROP field is not present in the Control message, then the default
                             01E8      ;       action is to use the previous value.
                             01E8      ;-
                             01E8
                             01E8      ROP_FIELD:                               ; Process record options field
                  03   E1    01E8              BBC     #DAP$V_ROP,-             ; Branch if ROP field was not explicitly
               44 A9         01EA                      DAP$W_CTLMENU(R9),-      ;  specified making previous ROP value
                  07         01EC                      KRF_FIELD                ;  the current value
      51       50 A9   D0    01ED              MOVL    DAP$L_ROP(R9),R1         ; Get DAP ROP bits
              02F7     30    01F1              BSBW    MAP_ROP_FIELD            ; Update ROP options in RAB
                             01F4
                             01F4      ;+
                             01F4      ; Process the DAP KRF field.
                             01F4      ; This field is applicable only for indexed files.
                             01F4      ;
                             01F4      ; Note: If the KRF field is not present in the Control message, then the default
                             01F4      ;       action is to use the previous value.
                             01F4      ;-
                             01F4
                             01F4      KRF_FIELD:                               ; Process key of reference field
                  02   E1    01F4              BBC     #DAP$V_KRF,-             ; Branch if KRF field was not explicitly
               44 A9         01F6                      DAP$W_CTLMENU(R9),-      ;  specified making previous KRF value
                  05         01F8                      KEY_FIELD                ;  the current value
               47 A9   90    01F9              MOVB    DAP$B_KRF(R9),-          ; Update key of reference value in RAB
               35 AB         01FC                      RAB$B_KRF(R11)           ;  (meaningful only for indexed files)
                             01FE
                             01FE      ;+
                             01FE      ; Process the DAP KEY field.
```

```
                           01FE  ; Its format and content are context dependent:
                           01FE  ;    (1) for block I/O access, it contains the virtual block number for
                           01FE  ;        $READ/$WRITE, or the number of blocks for $SPACE.
                           01FE  ;   (2a) for sequential record access without the key limit option in force,
                           01FE  ;        this field is ignored because RMS will use its internally stored
                           01FE  ;        next-record-pointer to locate the record.
                           01FE  ;   (2b) for sequential record access of an indexed file with the key limit
                           01FE  ;        option set (i.e., ROP = LIM), it contains the key value string.
                           01FE  ;   (3a) for random access by key value for relative (or fixed length
                           01FE  ;        sequential) files, it contains the relative record number.
                           01FE  ;   (3b) for random access by key value for indexed files, it contains the
                           01FE  ;        key value string.
                           01FE  ;    (4) for random access by record file address, it contains the RFA value.
                           01FE  ;-
                           01FE
                           01FE  KEY_FIELD:                                      ; Process the key field
          50   48 A9   7D  01FE          MOVQ    DAP$Q_KEY(R9),R0                ; <R0,R1> => descriptor of key field
          47 68    09  E0  0202          BBS     #FAL$V_BLK_IO,(R8),50$          ; Branch if block I/O access
                           0206
                           0206          ASSUME  RAB$C_SEQ EQ 0
                           0206          ASSUME  RAB$C_KEY EQ 1
                           0206          ASSUME  RAB$C_RFA EQ 2
                           0206
                           0206          $CASEB  SELECTOR=RAB$B_RAC(R11)-; Dispatch on RMS record access mode:
                           0206                  BASE=#RAB$C_SEQ-               ;
                           0206                  DISPL=<-                       ;
                           0206                      10$-                       ;  Sequential record access
                           0206                      20$-                       ;  Random access by key value
                           0206                      40$-                       ;  Random access by RFA
                           0206                  >
              01   E0      0211  10$:    BBS     #DAP$V_KEY,-                   ; Update key value only if KEY field
           1C 44 A9        0213                  DAP$W_CTLMENU(R9),30$          ;  was explicitly specified
              4D   11      0216                  BRB     90$                    ; All done with key field
       20   1D AA  91      0218  20$:    CMPB    FAB$B_ORG(R10),#FAB$C_IDX;Branch if indexed file
              14   13      021C          BEQL    30$                            ; Fall thru if sequential or relative
                           021E
                           021E  ;
                           021E  ; Key field contains a relative record number (RRN).
                           021E  ; RMS requires that the RRN be a 4-byte unsigned integer value.
                           021E  ;
                           021E
       34 AB   04  90      021E          MOVB    #4,RAB$B_KSZ(R11)              ; Store size and address of buffer
          01FC C8  DE      0222          MOVAL   FAL$L_NUMBER(R8),-             ;  that will hold RRN value
            30 AB          0226                  RAB$L_KBF(R11)                 ;  in KSZ/KBF fields of RAB
  04   00   61  50  2C     0228          MOVC5   R0,(R1),#0,#4,-                ; Copy RRN value as a longword to
          01FC C8          022D                  FAL$L_NUMBER(R8)               ;  buffer in FAL work area
              22   11      0230          BRB     60$                            ; Join common code
                           0232
                           0232  ;
                           0232  ; Key field contains a key string.
                           0232  ;
                           0232
       34 AB   50  90      0232  30$:    MOVB    R0,RAB$B_KSZ(R11)              ; Store size and address of buffer
          0700 C8  DE      0236          MOVAL   FAL$T_KEYBUF(R8),-             ;  that will hold key string value
            30 AB          023A                  RAB$L_KBF(R11)                 ;  in KSZ/KBF fields of RAB
  0700 C8   61  50  28     023C          MOVC3   R0,(R1),FAL$T_KEYBUF(R8); Copy string to buffer in FAL work area
              21   11      0242          BRB     90$                            ; All done with key field
```

```
                                0244
                                0244     ;
                                0244     ; Key field contains a record file address (RFA).
                                0244     ; RMS requires that the RFA be a 6-byte unsigned integer value.
                                0244     ;
                                0244
       06    00    61    50  2C 0244     40$:    MOVC5   R0,(R1),#0,#6,-         ; Store RFA value directly in RFA field
                      10 AB      0249                    RAB$W_RFA(R11)          ;  of RAB (zero extended to 6-bytes)
                         07  11 024B             BRB     60$                     ; Join common code
                                024D
                                024D     ;
                                024D     ; Key field contains virtual block number (VBN).
                                024D     ; RMS requires that the VBN be a 4-byte unsigned integer value.
                                024D     ;
                                024D
       04    00    61    50  2C 024D     50$:    MOVC5   R0,(R1),#0,#4,-         ; Store VBN value directly in BKT field
                      38 AB      0252                    RAB$L_BKT(R11)          ;  of RAB (zero extended to longword)
                                0254
                                0254     ;
                                0254     ; Common code to verify that the length of the string in the DAP KEY field
                                0254     ; does not exceed the size of the buffer used to store the string.
                                0254     ;
                                0254
                         0F  1B 0254     60$:    BLEQU   90$                     ; Done if all SRC bytes are copied
                                0256                                             ;  (i.e., SRC size LEQU DST size)
                         81  95 0256     70$:    TSTB    (R1)+                   ; Error if any unmoved bytes are
                         05  12 0258             BNEQ    80$                     ;  non-zero
                      F9 50  F5 025A             SOBGTR  R0,70$                  ; Continue until all extra bytes
                         06  11 025D             BRB     90$                     ;  are checked
                      FD9E'  30 025F     80$:    BSBW    FAL$UNS_KEY             ; Return error in Status message
                       0324  31 0262             BRW     EXIT_FAILURE            ; Exit state with failure
                                0265
                                0265     ;
                                0265     ; Use the CTLFUNC field value as the next state table value.
                                0265     ;
                                0265
                      40 A9  90 0265     90$:    MOVB    DAP$B_CTLFUNC(R9),-     ; Store new state transition value
                      10 A8      0268                    FAL$B_VALUE(R8)         ;
                       031F  31 026A             BRW     EXIT_SUCCESS            ; Exit state with success
```

```
                     026D                  .SBTTL  FAL$DECODE_CON
                     026D
                     026D      ;++
                     026D      ; Process the Continue Transfer message which has been received and validated.
                     026D      ;--
                     026D
                     026D      FAL$DECODE_CON::                               ; Entry point
                     026D
                     026D      ;
                     026D      ; Use the CONFUNC field value as the next state transition table value.
                     026D      ;
                     026D
    40 A9      90    026D              MOVB    DAP$B_CONFUNC(R9),-            ; Store new state transition value
    10 A8            0270                      FAL$B_VALUE(R8)                ;
     0317      31    0272              BRW     EXIT_SUCCESS                   ; Exit state with success
```

```
                     0275           .SBTTL  FAL$DECODE_CMP
                     0275
                     0275   ;++
                     0275   ; Process the Access Complete message which has been received and validated.
                     0275   ; Update the FAB if necessary.
                     0275   ;--
                     0275
                     0275   FAL$DECODE_CMP::                              ; Entry point
                     0275
                     0275   ;
                     0275   ; Process the DAP FOP field.
                     0275   ; Do not update the FOP field in the FAB if this is a DAP DISCONNECT function
                     0275   ; or if no FOP field was included in the Access Complete message.
                     0275   ;
                     0275
  51    44 A9    D0  0275           MOVL    DAP$L_FOP2(R9),R1         ; Get DAP FOP bits
           03    13  0279           BEQL    10$                       ; Branch if no bits to map
         01C6    30  027B           BSBW    MAP_FOP_FIELD             ; Update FOP in FAB
                     027E
                     027E   ;
                     027E   ; Use the CMPFUNC field value as the next state table value.
                     027E   ;
                     027E
        40 A9    90  027E   10$:    MOVB    DAP$B_CMPFUNC(R9),-       ; Store new state transition value
        10 A8        0281                   FAL$B_VALUE(R8)           ;
         0306    31  0283           BRW     EXIT_SUCCESS              ; Exit state with success
```

```
                      0286          .SBTTL  FAL$DECODE_KEY
                      0286
                      0286  ;++
                      0286  ; Process the Key Definition message which has been received and validated.
                      0286  ; Update the KEYXAB (by key of reference) with information from this message.
                      0286  ;--
                      0286
                      0286  FAL$DECODE_KEY::                              ; Entry point
                      0286
                      0286  ;
                      0286  ; Initialize the appropriate Key Definition XAB (in the FAL work area) and
                      0286  ; process the DAP REF field.
                      0286  ;
                      0286
      56   6C A9   9A 0286          MOVZBL  DAP$B_REF(R9),R6        ; Get key of reference value
            FD73'  30 028A          BSBW    FAL$INIT_KEYXAB         ; On return R7 = address of XAB
            03 50  E8 028D          BLBS    R0,10$                  ; Branch on success
             02F6  31 0290          BRW     EXIT_FAILURE            ; Exit state with failure
                      0293                                          ; (i.e., ignore this DAP message)
                      0293
                      0293  ;
                      0293  ; Process the DAP KNM field.
                      0293  ;
                      0293
                      0293          ASSUME  FAL$K_KEYNAM EQ 32
                      0293
           64 A9   D5 0293  10$:    TSTL    DAP$Q_KNM(R9)           ; Branch if no key name string was
              0B   13 0296          BEQL    20$                     ;  specified
           64 A9   2C 0298          MOVC5   DAP$Q_KNM(R9),-         ; Copy DAP key name string
           68 B9      029B                  @DAP$Q_KNM+4(R9),-      ;  to 32 byte key name buffer
38 B7   20    00      029D                  #0,#32,@XAB$L_KNM(R7)   ;  with zero fill
              03   11 02A1          BRB     30$                     ;
           38 A7   D4 02A3  20$:    CLRL    XAB$L_KNM(R7)           ; Zero key name buffer address
                      02A6
                      02A6  ;
                      02A6  ; Process the DAP FLG field.
                      02A6  ;
                      02A6
      51   48 A9   9A 02A6  30$:    MOVZBL  DAP$B_FLG(R9),R1        ; Get DAP FLG bits
              52   D4 02AA          CLRL    R2                      ; Clear RMS FLG bits
                      02AC          $MAPBIT DAP$V_DUP,XAB$V_DUP     ; Map DUP bit
                      02B4          $MAPBIT DAP$V_CHG,XAB$V_CHG     ; Map CHG bit
                      02BC          $MAPBIT DAP$V_NUL_CHR,XAB$V_NUL ; Map NUL bit
   12 A7      52   90 02C4          MOVB    R2,XAB$B_FLG(R7)        ; Update FLG field in XAB
                      02C8
                      02C8  ;
                      02C8  ; Process the DAP DFL, IFL, NUL, IAN, LAN, DAN, and DTP fields.
                      02C8  ;
                      02C8
   1C A7   44 A9   90 02C8          MOVB    DAP$W_DFL(R9),XAB$W_DFL(R7)
   1A A7   46 A9   90 02CD          MOVB    DAP$W_IFL(R9),XAB$W_IFL(R7)
   15 A7   6D A9   90 02D2          MOVB    DAP$B_NUL(R9),XAB$B_NUL(R7)
   08 A7   6E A9   90 02D7          MOVB    DAP$B_IAN(R9),XAB$B_IAN(R7)
   09 A7   6F A9   90 02DC          MOVB    DAP$B_LAN(R9),XAB$B_LAN(R7)
   0A A7   70 A9   90 02E1          MOVB    DAP$B_DAN(R9),XAB$B_DAN(R7)
   13 A7   71 A9   90 02E6          MOVB    DAP$B_DTP(R9),XAB$B_DTP(R7)
                      02EB
```

```
                          02EB  ;
                          02EB  ; Process the DAP NSG, POS, and SIZ fields.
                          02EB  ;
                          02EB  ; Note: FAL$DECODE_MSG guarantees that 0 < DAP$B_NSG < 9.
                          02EB  ;
                          02EB
          56   49 A9  9A  02EB         MOVZBL  DAP$B_NSG(R9),R6        ; Get # key segments
          14 A7   56  90  02EF         MOVB    R6,XAB$B_NSG(R7)        ; Update NSG field in XAB
          5C A9   56  28  02F3         MOVC3   R6,DAP$B_SIZ(R9),-      ; Copy 1 to 8 key size values
                2E A7     02F7                 XAB$B_SIZ(R7)           ;  to XAB
     56   56      01  78  02F9         ASHL    #1,R6,R6                ; Double byte count
          4C A9   56  28  02FD         MOVC3   R6,DAP$W_POS(R9),-      ; Copy 1 to 8 key position values
                1E A7     0301                 XAB$W_POS(R7)           ;  to XAB
                          0303
                          0303  ;
                          0303  ; Ignore the DAP RVB, DVB, DBS, IBS, LVL, TKS, and MRL fields as these are not
                          0303  ; inputs to RMS.
                          0303  ;
                          0303  ; Finish paper work and exit.
                          0303  ;
                          0303
                          0303         $SETBIT #FAL$V_KEYXAB,FAL$W_RECEIVED(R8)
                          0308                                         ; Denote XAB to add to XAB chain
             0281     31  0308         BRW     EXIT_SUCCESS            ; Exit state with success
```

```
                                030B          .SBTTL  FAL$DECODE_ALL
                                030B
                                030B    ;++
                                030B    ; Process the Allocation message which has been received and validated.
                                030B    ; Update the ALLXAB (by AID) with information from this message.
                                030B    ;--
                                030B
                                030B    FAL$DECODE_ALL::                                ; Entry point
                                030B
                                030B    ;
                                030B    ; Initialize the appropriate Allocation XAB (in the FAL work area) and
                                030B    ; process the DAP AID field.
                                030B    ;
                                030B
          56    50 A9      9A   030B          MOVZBL  DAP$B_AID(R9),R6        ; Get area ID value
                FCEE'      30   030F          BSBW    FAL$INIT_ALLXAB         ; On return R7 = address of XAB
                03 50      E8   0312          BLBS    R0,10$                  ; Branch on success
                 0271      31   0315          BRW     EXIT_FAILURE            ; Exit state with failure
                                0318                                          ; (i.e., ignore this DAP message)
                                0318
                                0318    ;
                                0318    ; Process the DAP ALN field.
                                0318    ;
                                0318
                                0318          ASSUME  DAP$K_ANY EQ 0
                                0318          ASSUME  DAP$K_CYL EQ XAB$C_CYL
                                0318          ASSUME  DAP$K_LBN EQ XAB$C_LBN
                                0318          ASSUME  DAP$K_VBN EQ XAB$C_VBN
                                0318
       09 A7    44 A9      90   0318    10$:  MOVB    DAP$B_ALN(R9),XAB$B_ALN(R7)
                                031D
                                031D    ;
                                031D    ; Process the DAP AOP field.
                                031D    ;
                                031D
          51    45 A9      9A   031D          MOVZBL  DAP$B_AOP(R9),R1        ; Get DAP AOP bits
                   52      D4   0321          CLRL    R2                      ; Clear RMS AOP bits
                                0323          $MAPBIT DAP$V_HRD,XAB$V_HRD     ; Map HRD bit
                                032B          $MAPBIT DAP$V_CBT2,XAB$V_CBT    ; Map CBT bit
                                0333          $MAPBIT DAP$V_CTG2,XAB$V_CTG    ; Map CTG bit
                                033B          $MAPBIT DAP$V_ONC,XAB$V_ONC     ; Map ONC bit
       08 A7       52      90   0343          MOVB    R2,XAB$B_AOP(R7)        ; Update AOP field in XAB
                                0347
                                0347    ;
                                0347    ; Process the DAP VOL, LOC, ALQ, BKZ, and DEQ fields.
                                0347    ;
                                0347
       0A A7    42 A9      B0   0347          MOVW    DAP$W_VOL(R9),XAB$W_VOL(R7)
       0C A7    48 A9      D0   034C          MOVL    DAP$L_LOC(R9),XAB$L_LOC(R7)
       10 A7    4C A9      D0   0351          MOVL    DAP$L_ALQ2(R9),XAB$L_ALQ(R7)
       16 A7    51 A9      90   0356          MOVB    DAP$B_BKZ(R9),XAB$B_BKZ(R7)
       14 A7    52 A9      B0   035B          MOVW    DAP$W_DEQ2(R9),XAB$W_DEQ(R7)
                                0360
                                0360    ;
                                0360    ; Finish paper work and exit.
                                0360    ;
                                0360
                   0360      $SETBIT #FAL$V_ALLXAB,FAL$W_RECEIVED(R8)
                   0365                                       ; Denote XAB to add to XAB chain
        0224   31  0365      BRW     EXIT_SUCCESS             ; Exit state with success
```

```
FALA
Psec

PSEC
----
.  A
$ABS
FAL$

Phas
----
Init
Comm
Pass
Symb
Pass
Symb
Psec
Cros
Asse

The
1112
Ther
1042
36 p

Macr
----
_$25
_$25
TOTA

2114

Ther

MACR
```

```
                        0368            .SBTTL  FAL$DECODE_TIM
                        0368
                        0368    ;++
                        0368    ; Process the Date and Time message which has been received and validated.
                        0368    ; Initialize both the DATXAB and RDTXAB and update them with information from
                        0368    ; this message. Other action routines will determine which of the two XABs to
                        0368    ; to use (or both) depending on the function that will be performed.
                        0368    ;--
                        0368
                        0368    FAL$DECODE_TIM::                                ; Entry point
                        0368
                        0368    ;
                        0368    ; Initialize and fill-in the Date and Time XAB.
                        0368    ;
                        0368
           FC95'   30   0368            BSBW    FAL$INIT_DATXAB         ; On return R7 = address of XAB
          48 A9    7D   036B            MOVQ    DAP$Q_CDT(R9),-         ; Copy creation date and time
          14 A7         036E                    XAB$Q_CDT(R7)           ;  binary value to XAB
          50 A9    7D   0370            MOVQ    DAP$Q_RDT(R9),-         ; Copy revision date and time
          0C A7         0373                    XAB$Q_RDT(R7)           ;  binary value to XAB
          58 A9    7D   0375            MOVQ    DAP$Q_EDT(R9),-         ; Copy expiration date and time
          1C A7         0378                    XAB$Q_EDT(R7)           ;  binary value to XAB
          60 A9    7D   037A            MOVQ    DAP$Q_BDT(R9),-         ; Copy backup date and time
          24 A7         037D                    XAB$Q_BDT(R7)           ;  binary value to XAB
          42 A9    B0   037F            MOVW    DAP$W_RVN(R9),-         ; Store revision number value in XAB
          08 A7         0382                    XAB$W_RVN(R7)           ;
                        0384
                        0384    ;
                        0384    ; Initialize and fill-in the Revision Date and Time XAB.
                        0384    ;
                        0384
           FC79'   30   0384            BSBW    FAL$INIT_RDTXAB         ; On return R7 = address of XAB
          50 A9    7D   0387            MOVQ    DAP$Q_RDT(R9),-         ; Copy revision date and time
          0C A7         038A                    XAB$Q_RDT(R7)           ;  binary value to XAB
          42 A9    B0   038C            MOVW    DAP$W_RVN(R9),-         ; Store revision number value in XAB
          08 A7         038F                    XAB$W_RVN(R7)           ;
                        0391
                        0391    ;
                        0391    ; Finish paper work and exit.
                        0391    ;
                        0391
   72 A8    14     A8   0391            BISW2   #<<FAL$M_DATXAB>!-      ; Denote XABs to add to XAB chain
                        0395                      <FAL$M_RDTXAB>!-      ;
                        0395                    0>,FAL$W_RECEIVED(R8)   ;
           01F4    31   0395            BRW     EXIT_SUCCESS            ; Exit state with success
```

```
                                                    J 11
FALACTMSG                         - STATE TABLE ACTION ROUTINES       16-SEP-1984 01:36:46  VAX/VMS Macro V04-00        Page  22
V04-000                           FAL$DECODE_PRO                       5-SEP-1984 01:16:21  [FAL.SRC]FALACTMSG.MAR;1         (13)

                            0398   848          .SBTTL  FAL$DECODE_PRO
                            0398   849
                            0398   850 ;++
                            0398   851 ; Process the Protection message which has been received and validated.
                            0398   852 ; Update the PROXAB with information from this message.
                            0398   853 ;--
                            0398   854
                            0398   855 FAL$DECODE_PRO::                          ; Entry point
               FC65'   30   0398   856          BSBW    FAL$INIT_PROXAB          ; On return R7 = address of XAB
               0C A7   D4   039B   857          CLRL    XAB$L_UIC(R7)            ; Initialize UIC and protection mask
  08 A7   FFFF 8F      B0   039E   858          MOVW    #-1,XAB$W_PRO(R7)        ;  fields to [0,0] and -1. These mean
                            03A4   859                                           ;  use process UIC and default process
                            03A4   860                                           ;  protection in effect, respectively
                            03A4   861
                            03A4   862 ;
                            03A4   863 ; Process the DAP OWNER field.
                            03A4   864 ;
                            03A4   865
       54    48 A9     7D   03A4   866          MOVQ    DAP$Q_OWNER(R9),R4       ; Get descriptor of ASCII string
         5B 8F   65    91   03A8   867          CMPB    (R5),#^A\[\              ; Branch if string does not begin
                 4C    12   03AC   868          BNEQ    30$                      ;  with bracket
  5D 8F     FF A544    91   03AE   869          CMPB    -1(R5)[R4],#^A\]\        ; Branch if string does not end
                 44    12   03B4   870          BNEQ    30$                      ;  with bracket
            54   02    C2   03B6   871          SUBL2   #2,R4                    ; Discard brackets
                 55    D6   03B9   872          INCL    R5                       ;
     65     54   2C    3A   03BB   873          LOCC    #^A\,\,R4,(R5)           ; Locate group-member delimiter
                 39    13   03BF   874          BEQL    30$                      ; Branch on failure
     54     51   55    C3   03C1   875          SUBL3   R5,R1,R4                 ; <R4,R5> => group string
                 50    D7   03C5   876          DECL    R0                       ; <R0,R1> => member string
                 51    D6   03C7   877          INCL    R1                       ;
                 7E    D4   03C9   878          CLRL    -(SP)                    ; Allocate space from stack
                 5E    DD   03CB   879          PUSHL   SP                       ; Address of result
                 51    DD   03CD   880          PUSHL   R1                       ; Address of input string
                 50    DD   03CF   881          PUSHL   R0                       ; Size of input string
  00000000'GF    03    FB   03D1   882          CALLS   #3,G^LIB$CVT_OTB         ; Convert octal string to binary
              1D 50    E9   03D8   883          BLBC    R0,20$                   ; Branch on failure
       0C A7     6E    B0   03DB   884          MOVW    (SP),XAB$W_MBM(R7)       ; Update member UIC value in XAB
                 5E    DD   03DF   885          PUSHL   SP                       ; Address of result
                 55    DD   03E1   886          PUSHL   R5                       ; Address of input string
                 54    DD   03E3   887          PUSHL   R4                       ; Size of input string
  00000000'GF    03    FB   03E5   888          CALLS   #3,G^LIB$CVT_OTB         ; Convert octal string to binary
              06 50    E9   03EC   889          BLBC    R0,10$                   ; Branch on failure
       0E A7     6E    B0   03EF   890          MOVW    (SP),XAB$W_GRP(R7)       ; Update group UIC value in XAB
                 03    11   03F3   891          BRB     20$                      ; UIC has been successfully converted
            0C   A7    B4   03F5   892 10$:     CLRW    XAB$W_MBM(R7)            ; GRP is invalid, so also discard MBM
                 8E    D4   03F8   893 20$:     CLRL    (SP)+                    ; Deallocate space from stack
                            03FA   894
                            03FA   895 ;
                            03FA   896 ; Process the DAP PROSYS, PROOWN, PROGRP, PROWLD fields.
                            03FA   897 ;
                            03FA   898
       40 A9     1E    B3   03FA   899 30$:     BITW    #<<DAP$M_PROSYS>!-       ; Use default file protection in effect
                            03FE   900                    <DAP$M_PROOWN>!-       ;  for the user process if all four
                            03FE   901                    <DAP$M_PROGRP>!-       ;  protection fields of the DAP
                            03FE   902                    <DAP$M_PROWLD>!-       ;  Protection message were defaulted
                            03FE   903                  0>,DAP$W_PROMENU(R9)     ;  (i.e., omitted from message)
                 1C    13   03FE   904          BEQL    40$                      ; Branch if no fields explicitly sent
```

```
      50    04    00    50 A9    F0  0400   905           INSV    DAP$W_PROSYS(R9),#0,#4,R0 ; Map system bits
      50    04    04    52 A9    F0  0406   906           INSV    DAP$W_PROOWN(R9),#4,#4,R0 ; Map owner bits
      50    04    08    54 A9    F0  040C   907           INSV    DAP$W_PROGRP(R9),#8,#4,R0 ; Map group bits
      50    04    0C    56 A9    F0  0412   908           INSV    DAP$W_PROWLD(R9),#12,#4,R0; Map world bits
                  08 A7     50   B0  0418   909           MOVW    R0,XAB$W_PRO(R7)          ; Update protection mask in XAB
                                     041C   910
                                     041C   911 ;
                                     041C   912 ; Finish paper work and exit.
                                     041C   913 ;
                                     041C   914
                                     041C   915 40$:      $SETBIT #FAL$V_PROXAB,FAL$W_RECEIVED(R8)
                                     0421   916                                             ; Denote XAB to add to XAB chain
                         0168    31  0421   917           BRW     EXIT_SUCCESS              ; Exit state with success
```

```
                               0424          .SBTTL  FAL$DECODE_NAM
                               0424
                               0424  ;++
                               0424  ; Process the name message which has been received and validated.
                               0424  ; Update FAB2 with information from this message.
                               0424  ;
                               0424  ; NOTE: At this time, only a rename operation will cause a Name message to be
                               0424  ; returned by FAL.
                               0424  ;--
                               0424
                               0424  FAL$DECODE_NAM::                            ; Entry point
 5A    0800 C8        DE       0424          MOVAL   FAL$L_FAB2(R8),R10          ; Put new filename FAB (FAB2) in R10
         44 A9        90       0429          MOVB    DAP$Q_NAMESPEC(R9),-        ; Store size of new filespec string
         34 AA                 042C                  FAB$B_FNS(R10)              ;  in FAB2
         44 A9        28       042E          MOVC3   DAP$Q_NAMESPEC(R9),-        ; Copy the filespec string to buffer
         48 B9                 0431                  @DAP$Q_NAMESPEC+4(R9),-     ;
         2C BA                 0433                  @FAB$L_FNA(R10)             ;
  52     44 A9        7E       0435          MOVAQ   DAP$Q_NAMESPEC(R9),R2       ; Get address of filename descriptor
            FBC4'     30       0439          BSBW    FAL$LOG_REQNAM2             ; Log requested new name in print file
 5A    0200 C8        DE       043C          MOVAL   FAL$L_FAB(R8),R10           ; Restore old filename FAB in R10
            0148      31       0441          BRW     EXIT_SUCCESS                ; Exit state with success
```

```
                        0444             .SBTTL  SUPPORT ROUTINES
                        0444
                        0444
                        0444             .SBTTL  MAP_FOP_FIELD
                        0444
                        0444     ;++
                        0444     ; This routine maps DAP FOP bits into RMS FOP bits and stores the result in
                        0444     ; the FOP field of the FAB.
                        0444     ;
                        0444     ; R1 contains the DAP bitmask on input.
                        0444     ; R2 is destroyed on output.
                        0444     ;--
                        0444
                        0444     MAP_FOP_FIELD:                                          ; Entry point
               52   D4  0444             CLRL    R2                                     ; Clear RMS FOP bits
               51   D5  0446             TSTL    R1                                     ; Examine FOP bitmask
               03   12  0448             BNEQ    10$                                    ; Begin mapping if any bits are set
             0090   31  044A             BRW     20$                                    ; Branch if there are no bits to map
                        044D     10$:    $MAPBIT DAP$V_RWO,FAB$V_RWO                    ; Map RWO bit
                        0455             $MAPBIT DAP$V_RWC,FAB$V_RWC                    ; Map RWC bit
                        045D             $MAPBIT DAP$V_POS,FAB$V_POS                    ; Map POS bit
                        0465             $MAPBIT DAP$V_CTG,FAB$V_CTG                    ; Map CTG bit
                        046D             $MAPBIT DAP$V_SUP,FAB$V_SUP                    ; Map SUP bit
                        0475             $MAPBIT DAP$V_NEF,FAB$V_NEF                    ; Map NEF bit
                        047D             $MAPBIT DAP$V_TMP,FAB$V_TMP                    ; Map TMP bit
                        0485             $MAPBIT DAP$V_TMD,FAB$V_TMD                    ; Map TMD bit
                        048D             $MAPBIT DAP$V_DMO,FAB$V_DMO                    ; Map DMO bit
                        0495             $MAPBIT DAP$V_WCK,FAB$V_WCK                    ; Map WCK bit
                        049D             $MAPBIT DAP$V_RCK,FAB$V_RCK                    ; Map RCK bit
                        04A5     ; ***** $MAPBIT DAP$V_CIF,FAB$V_CIF                    ; Map CIF bit
                        04A5             $MAPBIT DAP$V_SQO,FAB$V_SQO                    ; Map SQO bit
                        04AD             $MAPBIT DAP$V_MXV,FAB$V_MXV                    ; Map MXV bit
                        04B5             $MAPBIT DAP$V_SPL,FAB$V_SPL                    ; Map SPL bit
                        04BD             $MAPBIT DAP$V_SCF,FAB$V_SCF                    ; Map SCF bit
                        04C5             $MAPBIT DAP$V_DLT,FAB$V_DLT                    ; Map DLT bit
                        04CD             $MAPBIT DAP$V_CBT,FAB$V_CBT                    ; Map CBT bit
                        04D5     ; ***** $MAPBIT DAP$V_DFW,FAB$V_DFW                    ; Map DFW bit
                        04D5             $MAPBIT DAP$V_TEF,FAB$V_TEF                    ; Map TEF bit
                        04DD     ;       $MAPBIT DAP$V_OFP,FAB$V_OFP                    ; Map OFP bit
                        04DD                                                            ; Note: this bit has no meaning here
                        04DD                                                            ;  because only primary filespec
                        04DD                                                            ;  is being given to RMS by FAL
   04 04 AA    18   E1  04DD     20$:    BBC     #FAB$V_NAM,FAB$L_FOP(R10),30$
                        04E2             $SETBIT #FAB$V_NAM,R2                          ; Preserve state of NAM bit in FOP
      04 AA    52   D0  04E6     30$:    MOVL    R2,FAB$L_FOP(R10)                      ; Update FOP field in FAB
                    05  04EA             RSB                                            ; Exit
```

```
                        04EB         .SBTTL  MAP_ROP_FIELD
                        04EB
                        04EB   ;+
                        04EB   ; This routine maps DAP ROP bits into RMS ROP bits and stores the result in
                        04EB   ; the ROP field of the RAB.
                        04EB   ;
                        04EB   ; R1 contains the DAP bitmask on input.
                        04EB   ; R2 is destroyed on output.
                        04EB   ;-
                        04EB
                        04EB   MAP_ROP_FIELD:                                  ; Entry point
                52   D4 04EB         CLRL    R2                          ; Clear RMS ROP bits
                51   D5 04ED         TSTL    R1                          ; Examine ROP bitmask
                03   12 04EF         BNEQ    10$                         ; Begin mapping if any bits are set
              0090   31 04F1         BRW     20$                         ; Branch if there are no bits to map
                        04F4   10$:  $MAPBIT DAP$V_EOF,RAB$V_EOF         ; Map EOF bit
                        04FC         $MAPBIT DAP$V_FDL,RAB$V_FDL         ; Map FDL bit
                        0504         $MAPBIT DAP$V_UIF,RAB$V_UIF         ; Map UIF bit
                        050C         $MAPBIT DAP$V_LOA,RAB$V_LOA         ; Map LOA bit
                        0514         $MAPBIT DAP$V_ULK,RAB$V_ULK         ; Map ULK bit
                        051C         $MAPBIT DAP$V_TPT,RAB$V_TPT         ; Map TPT bit
                        0524         $MAPBIT DAP$V_RAH,RAB$V_RAH         ; Map RAH bit
                        052C         $MAPBIT DAP$V_WBH,RAB$V_WBH         ; Map WBH bit
                        0534         $MAPBIT DAP$V_KGE,RAB$V_KGE         ; Map KGE bit
                        053C         $MAPBIT DAP$V_KGT,RAB$V_KGT         ; Map KGT bit
                        0544         $MAPBIT DAP$V_NLK,RAB$V_NLK         ; Map NLK bit
                        054C         $MAPBIT DAP$V_RLK,RAB$V_RLK         ; Map RLK bit
                        0554         $MAPBIT DAP$V_ROPBIO,RAB$V_BIO      ; Map BIO bit
                        055C         $MAPBIT DAP$V_LIM,RAB$V_LIM         ; Map LIM bit
                        0564         $MAPBIT DAP$V_NXR,RAB$V_NXR         ; Map NXR bit
                        056C         $MAPBIT DAP$V_ROPWAT,RAB$V_WAT      ; Map WAT bit
                        0574         $MAPBIT DAP$V_RRL,RAB$V_RRL         ; Map RRL bit
                        057C         $MAPBIT DAP$V_REA,RAB$V_REA         ; Map REA bit
          04 AB   52 D0 0584   20$:  MOVL    R2,RAB$L_ROP(R11)           ; Update ROP field in RAB
                      05 0588         RSB                                 ; Exit
```

```
                         0589             .SBTTL  STATE EXIT ROUTINES
                         0589
                         0589    ;++
                         0589    ; Exit state with failure.
                         0589    ;--
                         0589
                         0589    EXIT_FAILURE:                                   ; Entry point
                 50  D4  0589             CLRL    R0                             ; Signal state transition failure
                     05  058B             RSB                                    ; Exit to state table manager
                         058C
                         058C    ;++
                         058C    ; Exit state with success.
                         058C    ;--
                         058C
                         058C    EXIT_SUCCESS:                                   ; Entry point
             50  01  D0  058C             MOVL    #1,R0                          ; Signal state transition success
                     05  058F             RSB                                    ; Exit to state table manager
                         0590
                         0590             .END                                   ; End of module
```

```
$$COUNT                         = 00000003            DAP$K_BLK_VBN                   = 00000004
DAP$B_ACCFUNC                     00000040            DAP$K_BLN                         000000C0
DAP$B_ACCOPT                      00000041            DAP$K_CNF_MSG                   = 00000001
DAP$B_AID                         00000050            DAP$K_CYL                       = 00000001
DAP$B_ALN                         00000044            DAP$K_DECVER_V                  = 00000004
DAP$B_AOP                         00000045            DAP$K_ECONUM_V                  = 00000000
DAP$B_BITCNT                      00000035            DAP$K_FIX                       = 00000001
DAP$B_BKS                         00000050            DAP$K_IDX                       = 00000020
DAP$B_BKZ                         00000051            DAP$K_KEY_ACC                   = 00000001
DAP$B_BLKCNT                      00000056            DAP$K_LBN                       = 00000002
DAP$B_BSZ                         00000052            DAP$K_LOAD                      = 000000FF
DAP$B_CMPFUNC                     00000040            DAP$K_REL                       = 00000010
DAP$B_CONFUNC                     00000040            DAP$K_RFA_ACC                   = 00000002
DAP$B_CTLFUNC                     00000040            DAP$K_RMS32                     = 00000003
DAP$B_DAN                         00000070            DAP$K_SEQ                       = 00000000
DAP$B_DATATYPE                    00000044            DAP$K_SEQ_ACC                   = 00000000
DAP$B_DBS                         0000007C            DAP$K_SEQ_FILE                  = 00000003
DAP$B_DCODE_FID                   00000019            DAP$K_STG                       = 00000000
DAP$B_DCODE_MAC                   0000001B            DAP$K_STM                       = 00000004
DAP$B_DCODE_MSG                   0000001A            DAP$K_STMCR                     = 00000006
DAP$B_DECVER                      00000047            DAP$K_STMLF                     = 00000005
DAP$B_DTP                         00000071            DAP$K_SYSCAP1_V                 = EFF67DF7
DAP$B_ECONUM                      00000045            DAP$K_SYSCAP2_V                 = 00001962
DAP$B_FAC                         00000042            DAP$K_UDF                       = 00000000
DAP$B_FILESYS                     00000043            DAP$K_USRNUM_V                  = 00000000
DAP$B_FLAGS                       00000031            DAP$K_USRVER_V                  = 00000000
DAP$B_FLG                         00000048            DAP$K_VAR                       = 00000002
DAP$B_FSZ                         00000051            DAP$K_VAXVMS                    = 00000007
DAP$B_IAN                         0000006E            DAP$K_VBN                       = 00000003
DAP$B_IBS                         0000007D            DAP$K_VERNUM_V                  = 00000007
DAP$B_KRF                         00000047            DAP$K_VFC                       = 00000003
DAP$B_LAN                         0000006F            DAP$L_ALQ1                        0000004C
DAP$B_LEN256                      00000034            DAP$L_ALQ2                        0000004C
DAP$B_LENGTH                      00000033            DAP$L_ATTMENU                     00000040
DAP$B_LVL                         0000007E            DAP$L_CMWA                        00000030
DAP$B_NAMETYPE                    00000040            DAP$L_CRC_RSLT                    00000020
DAP$B_NSG                         00000049            DAP$L_DCODE_STS                   00000018
DAP$B_NUL                         0000006D            DAP$L_DEV                         00000068
DAP$B_ORG                         00000045            DAP$L_DVB                         00000078
DAP$B_OSTYPE                      00000042            DAP$L_EBK                         00000078
DAP$B_RAC                         00000046            DAP$L_FOP1                        00000064
DAP$B_RAT                         00000047            DAP$L_FOP2                        00000044
DAP$B_REF                         0000006C            DAP$L_HBK                         00000074
DAP$B_RFM                         00000046            DAP$L_KEYMENU                     00000040
DAP$B_SHR                         00000043            DAP$L_LOC                         00000048
DAP$B_SIZ                         0000005C            DAP$L_MRN                         00000058
DAP$B_SIZ_TMP                     0000004A            DAP$L_MSG_MASK                    0000001C
DAP$B_STREAMID                    00000032            DAP$L_ROP                         00000050
DAP$B_TKS                         0000007F            DAP$L_RVB                         00000074
DAP$B_TYPE                        00000030            DAP$L_SBN                         0000007C
DAP$B_USRNUM                      00000046            DAP$L_SSPWA                       00000080
DAP$B_USRVER                      00000048            DAP$L_SSP_CAP                     00000088
DAP$B_VERNUM                      00000044            DAP$L_SSP_FLG                     00000084
DAP$B_X_FIELD                     00000024            DAP$L_TEMP                        00000090
DAP$C_BLN                         000000C0            DAP$M_BITCNT                    = 00000008
DAP$K_ANY                       = 00000000            DAP$M_BLKCNT                    = 00000040
DAP$K_BLK_FILE                  = 00000005            DAP$M_CMPFMT                    = 00000008
```

```
                                                                       D 12
FALACTMSG                                   - STATE TABLE ACTION ROUTINES          16-SEP-1984 01:36:46  VAX/VMS Macro V04-00        Page  29
Symbol table                                                                        5-SEP-1984 01:16:21  [FAL.SRC]FALACTMSG.MAR;1        (17)

DAP$M_DFTSPEC                    = 00000010             DAP$V_FDL                        = 00000001
DAP$M_DMO                        = 00002000             DAP$V_FTN                        = 00000000
DAP$M_DSP_3NAM                   = 00000200             DAP$V_GET                        = 00000001
DAP$M_EMBEDDED                   = 00000010             DAP$V_HRD                        = 00000000
DAP$M_GET                        = 00000002             DAP$V_KEY                        = 00000001
DAP$M_GO_NOGO                    = 00000010             DAP$V_KGE                        = 00000009
DAP$M_IMAGE                      = 00000002             DAP$V_KGT                        = 0000000A
DAP$M_LOADIM                     = 00000001             DAP$V_KRF                        = 00000002
DAP$M_LSA                        = 00000040             DAP$V_LIM                        = 0000000E
DAP$M_MACY11                     = 00000080             DAP$V_LOA                        = 00000004
DAP$M_MSE                        = 00000010             DAP$V_LOAD                       = 00000000
DAP$M_PROGRP                     = 00000008             DAP$V_MSGBLK                     = 00000012
DAP$M_PROOWN                     = 00000004             DAP$V_MXV                        = 00000013
DAP$M_PROSYS                     = 00000002             DAP$V_NEF                        = 00000009
DAP$M_PROWLD                     = 00000010             DAP$V_NIL                        = 00000006
DAP$M_SEGMENT                    = 00000040             DAP$V_NLK                        = 0000000B
DAP$M_TMP1$                      = 00000020             DAP$V_NUL_CHR                    = 00000002
DAP$M_TMP2$                      = 000000C0             DAP$V_NXR                        = 0000000F
DAP$M_TMP3$                      = 00020000             DAP$V_ONC                        = 00000003
DAP$M_TMP4$                      = 01000000             DAP$V_POS                        = 00000003
DAP$M_TMP5$                      = F0000000             DAP$V_PRN                        = 00000002
DAP$M_ZERO                       = 00000080             DAP$V_PUT                        = 00000000
DAP$Q_ADT                          00000070             DAP$V_RAC                        = 00000000
DAP$Q_BDT                          00000060             DAP$V_RAH                        = 00000007
DAP$Q_CDT                          00000048             DAP$V_RCK                        = 0000000F
DAP$Q_DCODE_FLG                    00000000             DAP$V_REA                        = 00000012
DAP$Q_EDT                          00000058             DAP$V_RLK                        = 0000000C
DAP$Q_FILESPEC                     00000044             DAP$V_ROP                        = 00000003
DAP$Q_KEY                          00000048             DAP$V_ROPBIO                     = 0000000D
DAP$Q_KNM                          00000064             DAP$V_ROPWAT                     = 00000010
DAP$Q_MSG_BUF1                     00000008             DAP$V_RRL                        = 00000011
DAP$Q_MSG_BUF2                     00000010             DAP$V_RWC                        = 00000001
DAP$Q_NAMESPEC                     00000044             DAP$V_RWO                        = 00000000
DAP$Q_OWNER                        00000048             DAP$V_SCF                        = 00000015
DAP$Q_PASSWORD                     00000050             DAP$V_SHRDEL                     = 00000002
DAP$Q_PDT                          00000068             DAP$V_SHRGET                     = 00000001
DAP$Q_RDT                          00000050             DAP$V_SHRPUT                     = 00000000
DAP$Q_RUNSYS                       0000005C             DAP$V_SHRUPD                     = 00000003
DAP$Q_SYSCAP                       00000028             DAP$V_SPL                        = 00000014
DAP$Q_SYSPEC                       00000038             DAP$V_SQO                        = 00000012
DAP$V_APP                        = 00000007             DAP$V_SUP                        = 00000008
DAP$V_BIGBLK                     = 00000014             DAP$V_TEF                        = 0000001A
DAP$V_BIO                        = 00000005             DAP$V_TMD                        = 0000000B
DAP$V_BLK                        = 00000003             DAP$V_TMP                        = 0000000A
DAP$V_BRO                        = 00000006             DAP$V_TPT                        = 00000006
DAP$V_CBT                        = 00000017             DAP$V_TRN                        = 00000004
DAP$V_CBT2                       = 00000002             DAP$V_UIF                        = 00000002
DAP$V_CHG                        = 00000001             DAP$V_ULK                        = 00000005
DAP$V_CR                         = 00000001             DAP$V_UPD                        = 00000003
DAP$V_CTG                        = 00000007             DAP$V_UPI                        = 00000005
DAP$V_CTG2                       = 00000001             DAP$V_VAXVMS                     = 00000034
DAP$V_DAPCRC                     = 00000015             DAP$V_WBH                        = 00000008
DAP$V_DEL                        = 00000002             DAP$V_WCK                        = 0000000E
DAP$V_DLT                        = 00000016             DAP$W_ALLMENU                      00000040
DAP$V_DMO                        = 0000000D             DAP$W_BLS                          00000048
DAP$V_DUP                        = 00000000             DAP$W_BUFSIZ                       00000040
DAP$V_EOF                        = 00000000             DAP$W_CHECK                        00000042
```

```
DAP$W_CTLMENU                  00000044                 FAB$V_DMO                    = 0000000C
DAP$W_DEQ1                     00000054                 FAB$V_EXE                    = 00000007
DAP$W_DEQ2                     00000052                 FAB$V_FTN                    = 00000000
DAP$W_DFL                      00000044                 FAB$V_GET                    = 00000001
DAP$W_DISPLAY1                 0000004C                 FAB$V_MXV                    = 00000001
DAP$W_DISPLAY2                 00000054                 FAB$V_NAM                    = 00000018
DAP$W_FFB                      00000072                 FAB$V_NEF                    = 0000000A
DAP$W_IFL                      00000046                 FAB$V_NIL                    = 00000005
DAP$W_LRL                      00000070                 FAB$V_POS                    = 00000008
DAP$W_MRL                      00000072                 FAB$V_PRN                    = 00000002
DAP$W_MRS                      0000004A                 FAB$V_PUT                    = 00000000
DAP$W_PARTNER                  00000006                 FAB$V_RCK                    = 00000017
DAP$W_POS                      0000004C                 FAB$V_RWC                    = 0000000B
DAP$W_POS_TMP                  0000004A                 FAB$V_RWO                    = 00000007
DAP$W_PROGRP                   00000054                 FAB$V_SCF                    = 0000000E
DAP$W_PROMENU                  00000040                 FAB$V_SHRDEL                 = 00000002
DAP$W_PROOWN                   00000052                 FAB$V_SHRGET                 = 00000001
DAP$W_PROSYS                   00000050                 FAB$V_SHRPUT                 = 00000000
DAP$W_PROWLD                   00000056                 FAB$V_SHRUPD                 = 00000003
DAP$W_RVN                      00000042                 FAB$V_SPL                    = 0000000D
DAP$W_SSP_MENU                 00000080                 FAB$V_SQO                    = 00000006
DAP$W_TIMENU                   00000040                 FAB$V_SUP                    = 00000002
DAP$W_VERSION                  00000004                 FAB$V_TEF                    = 0000001C
DAP$W_VOL                      00000042                 FAB$V_TMD                    = 00000004
EXIT_FAILURE                   00000589 R      02       FAB$V_TMP                    = 00000003
EXIT_SUCCESS                   0000058C R      02       FAB$V_TRN                    = 00000004
FAB$B_BKS                    = 0000003E                 FAB$V_UPD                    = 00000003
FAB$B_FAC                    = 00000016                 FAB$V_UPI                    = 00000006
FAB$B_FNS                    = 00000034                 FAB$V_WCK                    = 00000009
FAB$B_FSZ                    = 0000003F                 FAB$W_BLS                    = 0000003C
FAB$B_ORG                    = 0000001D                 FAB$W_DEQ                    = 00000014
FAB$B_RAT                    = 0000001E                 FAB$W_MRS                    = 00000036
FAB$B_RFM                    = 0000001F                 FAL$BUILD_HEAD                 ********   X    02
FAB$B_SHR                    = 00000017                 FAL$BUILD_TAIL                 ********   X    02
FAB$C_FIX                    = 00000001                 FAL$B_ACCFUNC                  000001F6
FAB$C_IDX                    = 00000020                 FAL$B_ACCOPT                   000001F5
FAB$C_REL                    = 00000010                 FAL$B_DATATYPE                 000001F4
FAB$C_SEQ                    = 00000000                 FAL$B_DISABLE                  00000006
FAB$C_STM                    = 00000004                 FAL$B_ENABLE                   00000005
FAB$C_STMCR                  = 00000006                 FAL$B_LOGGING                  00000004
FAB$C_STMLF                  = 00000005                 FAL$B_MISCOPT                  00000007
FAB$C_UDF                    = 00000000                 FAL$B_RAC                      000001F7
FAB$C_VAR                    = 00000002                 FAL$B_RBK_CACHE                00000012
FAB$C_VFC                    = 00000003                 FAL$B_RCVBUFIDX                00000011
FAB$L_ALQ                    = 00000010                 FAL$B_VALUE                    00000010
FAB$L_FNA                    = 0000002C                 FAL$CVT_BN8_EXT                ********   X    02
FAB$L_FOP                    = 00000004                 FAL$C_WRKBLN                   00002000
FAB$L_MRN                    = 00000038                 FAL$DECODE_ACC                 0000010B RG      02
FAB$M_CR                     = 00000002                 FAL$DECODE_ALL                 0000030B RG      02
FAB$V_BIO                    = 00000005                 FAL$DECODE_ATT                 000000A8 RG      02
FAB$V_BLK                    = 00000003                 FAL$DECODE_CMP                 00000275 RG      02
FAB$V_BRO                    = 00000006                 FAL$DECODE_CNF                 00000000 RG      02
FAB$V_CBT                    = 00000015                 FAL$DECODE_CON                 0000026D RG      02
FAB$V_CR                     = 00000001                 FAL$DECODE_CTL                 00000196 RG      02
FAB$V_CTG                    = 00000014                 FAL$DECODE_KEY                 00000286 RG      02
FAB$V_DEL                    = 00000002                 FAL$DECODE_NAM                 00000424 RG      02
FAB$V_DLT                    = 0000000F                 FAL$DECODE_PRO                 00000398 RG      02
```

```
                                                          F 12
FALACTMSG                       - STATE TABLE ACTION ROUTINES          16-SEP-1984 01:36:46  VAX/VMS Macro V04-00           Page  31
Symbol table                                                            5-SEP-1984 01:16:21  [FAL.SRC]FALACTMSG.MAR;1            (17)

FAL$DECODE_TIM                   00000368 RG    02      FAL$T_FALLOG                     00001C00
FAL$INIT_ACLXAB                  ********   X   02      FAL$T_FILESPEC                   00000400
FAL$INIT_DATXAB                  ********   X   02      FAL$T_FILESPEC2                  00000900
FAL$INIT_KEYXAB                  ********   X   02      FAL$T_KEYBUF                     00000700
FAL$INIT_PROXAB                  ********   X   02      FAL$T_MBXBUF                     00001980
FAL$INIT_RDTXAB                  ********   X   02      FAL$T_PRTBUF1                    00001A00
FAL$K_KEYNAM                   = 00000020               FAL$T_PRTBUF2                    00001B00
FAL$K_WRKBLN                     00002000               FAL$T_RESULT                     00000600
FAL$LOG_REQNAM                   ********   X   02      FAL$T_RESULT2                    00000B00
FAL$LOG_REQNAM2                  ********   X   02      FAL$T_SYSNET                     00001D00
FAL$L_ACLXAB                     00000C00               FAL$T_VOLNAME                    00001E00
FAL$L_ALLXABINI                  00000074               FAL$UNS_KEY                      ********   X   02
FAL$L_CHAIN_NXT                  0000007C               FAL$V_ACLXAB                   = 00000001
FAL$L_DATXAB                     00000320               FAL$V_ATT_MSG                  = 00000001
FAL$L_FAB                        00000200               FAL$V_BLK_IO                   = 00000009
FAL$L_FAB2                       00000800               FAL$V_CNF_MSG                  = 00000000
FAL$L_FHCXAB                     000002F4               FAL$V_DIS_CRC                  = 00000030
FAL$L_FOP                        000001F8               FAL$V_DIS_MBK                  = 00000031
FAL$L_KEYNAM                     00001C00               FAL$V_FTM                      = 00000008
FAL$L_KEYXAB                     00001000               FAL$V_KEYXAB                   = 00000000
FAL$L_KEYXABINI                  00000078               FAL$V_LAST_MSG                 = 00000018
FAL$L_NAM                        00000294               FAL$V_PROXAB                   = 00000003
FAL$L_NAM2                       00000850               FAL$V_USE_DBS                  = 00000039
FAL$L_NUMBER                     000001FC               FAL$V_USE_SC1                  = 0000003C
FAL$L_PROXAB                     0000034C               FAL$V_USE_SC2                  = 0000003D
FAL$L_RAB                        00000250               FAL$V_USE_SYS                  = 0000003A
FAL$L_RCVBUF                     0000005C               FAL$V_USE_VER                  = 0000003B
FAL$L_RDTXAB                     000003B0               FAL$V_WILD                     = 0000000A
FAL$L_RMS_PTR                    0000006C               FAL$W_DAPBUFSIZ                  0000001A
FAL$L_STB                        000000C0               FAL$W_DISPLAY                    00000070
FAL$L_SUMXAB                     000003A4               FAL$W_LNKCHN                     0000001C
FAL$L_TEMP                       000003F4               FAL$W_MBXCHN                     0000001E
FAL$L_USE_SC1                    000000A8               FAL$W_QIOBUFSIZ                  00000018
FAL$L_USE_SC2                    000000AC               FAL$W_RECEIVED                   00000072
FAL$L_USE_VER                    000000A4               FAL$W_USE_DBS                    000000A0
FAL$M_DATXAB                   = 00000004               FAL$W_USE_SYS                    000000A2
FAL$M_RDTXAB                   = 00000010               KEY_FIELD                        000001FE R     02
FAL$Q_BLD                        00000050               KRF_FIELD                        000001F4 R     02
FAL$Q_DIRNAME                    00000088               LIB$CVT_OTB                      ********   X   02
FAL$Q_FALLOG                     00000090               MAP_FOP_FIELD                    00000444 R     02
FAL$Q_FLG                        00000000               MAP_ROP_FIELD                    000004EB R     02
FAL$Q_MBX                        00000038               RAB$B_KRF                      = 00000035
FAL$Q_MBXIOSB                    00000030               RAB$B_KSZ                      = 00000034
FAL$Q_RCV                        00000040               RAB$B_RAC                      = 0000001E
FAL$Q_RCVIOSB                    00000020               RAB$C_KEY                      = 00000001
FAL$Q_RMS                        00000064               RAB$C_RFA                      = 00000002
FAL$Q_STATE_CTX                  00000008               RAB$C_SEQ                      = 00000000
FAL$Q_SYSNET                     00000098               RAB$L_BKT                      = 00000038
FAL$Q_TEMP                       000003F8               RAB$L_KBF                      = 00000030
FAL$Q_VOLNAME                    00000080               RAB$L_ROP                      = 00000004
FAL$Q_XMT                        00000048               RAB$V_BIO                      = 0000000B
FAL$Q_XMTIOSB                    00000028               RAB$V_EOF                      = 00000008
FAL$TRANSMIT                     ********   X   02      RAB$V_FDL                      = 00000006
FAL$T_DAP                        00000100               RAB$V_KGE                      = 00000015
FAL$T_DIRNAME                    00001F00               RAB$V_KGT                      = 00000016
FAL$T_EXPAND                     00000500               RAB$V_LIM                      = 0000000E
FAL$T_EXPAND2                    00000A00               RAB$V_LOA                      = 0000000D
```

```
RAB$V_NLK                        = 00000014
RAB$V_NXR                        = 00000017
RAB$V_RAH                        = 00000009
RAB$V_REA                        = 00000002
RAB$V_RLK                        = 00000013
RAB$V_RRL                        = 00000003
RAB$V_TPT                        = 00000001
RAB$V_UIF                        = 00000004
RAB$V_ULK                        = 00000012
RAB$V_WAT                        = 00000011
RAB$V_WBH                        = 0000000A
RAB$W_RFA                        = 00000010
RAC_FIELD                          0000019F R      02
ROP_FIELD                          000001E8 R      02
SEND_CNF                           00000015 R      02
XAB$B_ALN                        = 00000009
XAB$B_AOP                        = 00000008
XAB$B_BKZ                        = 00000016
XAB$B_DAN                        = 0000000A
XAB$B_DTP                        = 00000013
XAB$B_FLG                        = 00000012
XAB$B_IAN                        = 00000008
XAB$B_LAN                        = 00000009
XAB$B_NSG                        = 00000014
XAB$B_NUL                        = 00000015
XAB$B_SIZ                        = 0000002E
XAB$C_CYL                        = 00000001
XAB$C_LBN                        = 00000002
XAB$C_VBN                        = 00000003
XAB$L_ALQ                        = 00000010
XAB$L_KNM                        = 00000038
XAB$L_LOC                        = 0000000C
XAB$L_UIC                        = 0000000C
XAB$Q_BDT                        = 00000024
XAB$Q_CDT                        = 00000014
XAB$Q_EDT                        = 0000001C
XAB$Q_RDT                        = 0000000C
XAB$V_CBT                        = 00000005
XAB$V_CHG                        = 00000001
XAB$V_CTG                        = 00000007
XAB$V_DUP                        = 00000000
XAB$V_HRD                        = 00000000
XAB$V_NUL                        = 00000002
XAB$V_ONC                        = 00000001
XAB$W_DEQ                        = 00000014
XAB$W_DFL                        = 0000001C
XAB$W_GRP                        = 0000000E
XAB$W_IFL                        = 0000001A
XAB$W_LRL                          ********     X  02
XAB$W_MBM                        = 0000000C
XAB$W_POS                        = 0000001E
XAB$W_PRO                        = 00000008
XAB$W_RVN                        = 00000008
XAB$W_VOL                        = 0000000A
```

```
                                             +----------------+
                                             ! Psect synopsis !
                                             +----------------+

PSECT name                     Allocation          PSECT No.  Attributes
----------                     ----------          ---------  ----------
.  ABS  .                      00000000  (     0.)  00 (  0.)  NOPIC   USR   CON   ABS   LCL NOSHR NOEXE NORD   NOWRT NOVEC BYTE
$ABS$                          00002000  (  8192.)  01 (  1.)  NOPIC   USR   CON   ABS   LCL NOSHR   EXE   RD     WRT NOVEC BYTE
FAL$CODE                       00000590  (  1424.)  02 (  2.)  NOPIC   USR   CON   REL   LCL NOSHR   EXE   RD   NOWRT NOVEC BYTE

                                         +------------------------+
                                         ! Performance indicators !
                                         +------------------------+

Phase                    Page faults    CPU Time       Elapsed Time
-----                    -----------    --------       ------------
Initialization                    37    00:00:00.05    00:00:01.61
Command processing               142    00:00:00.36    00:00:02.39
Pass 1                           450    00:00:13.74    00:00:51.94
Symbol table sort                  0    00:00:01.65    00:00:04.98
Pass 2                           192    00:00:02.86    00:00:10.29
Symbol table output               59    00:00:00.30    00:00:00.60
Psect synopsis output              1    00:00:00.03    00:00:00.73
Cross-reference output             0    00:00:00.00    00:00:00.00
Assembler run totals             883    00:00:19.00    00:01:12.55

The working set limit was 1950 pages.
111221 bytes (218 pages) of virtual memory were used to buffer the intermediate code.
There were 90 pages of symbol table space allocated to hold 1637 non-local and 110 local symbols.
1042 source lines were read in Pass 1, producing 17 object records in Pass 2.
36 pages of virtual memory were used to define 35 macros.

                                        +--------------------------+
                                        ! Macro library statistics !
                                        +--------------------------+

Macro library name                          Macros defined
------------------                          --------------
_$255$DUA28:[FAL.OBJ]FAL.MLB;1                    20
_$255$DUA28:[SYSLIB]STARLET.MLB;2                 12
TOTALS (all libraries)                            32

2114 GETS were required to define 32 macros.

There were no errors, warnings or information messages.

MACRO/LIS=LIS$:FALACTMSG/OBJ=OBJ$:FALACTMSG MSRC$:FALACTMSG/UPDATE=(ENH$:FALACTMSG)+LIB$:FAL/LIB
```

0174 AH-BT13A-SE
VAX/VMS V4.0

DIGITAL EQUIPMENT CORPORATION
CONFIDENTIAL AND PROPRIETARY